Origin of anti-sickling activity via QSAR modelling

Chuleeporn Phanus-umporn, Saw Simeon, Watshara Shoombuatong and Chanin Nantasenamat

July 7, 2016

Reading the data file, removing the the column having Standard Deviation equal to 0

```
df <- read.csv("data.csv")
Activity <- df$Activity
descriptors <- df[, 2:ncol(df)]
curated_descriptors <- descriptors[, which(!apply(descriptors, 2, sd) == 0)]
real_data <- cbind(Activity, curated_descriptors)
```

Fuction for building classification model using random forest 100 times (over-sampling, stratified data splitting, stastical assessment including accuracy, sensitivity, specificity and matthew correlation coefficient)

```
RF_training_classification <- function(x) {
  library(parallel)
  library(doSNOW)
  cl <- makeCluster(8)
  registerDoSNOW(cl)

  ok <- list(100)
  ok <- foreach(i = 1:100 ) %dopar% {

    data <- x
    active <- subset(data, Activity == "active")
    inactive <- subset(data, Activity == "inactive")
    active <- dplyr::sample_n(active, size =  83, replace = TRUE)
    data <- rbind(active, inactive)
    trainIndex <- sample(nrow(data), size = as.integer(nrow(data) * 0.8),
                         replace = FALSE)
    train <- data[trainIndex, ]
    test <- data[-trainIndex, ]
    model_train <- ranger::ranger(Activity~., data = train, write.forest = TRUE, save.memory = TRUE)
    rm(ctrl)
    rm(rf)
    rm(data)
    rm(trainIndex)
    actual <- train$Activity
    prediction <- predict(model_train, train)
    prediction <- prediction$predictions
    rm(train)
    rm(test)
    rm(model_train)
    results <- caret::confusionMatrix(prediction, actual)
```

```
    results <- results$table
    results <- as.numeric(results)
    rm(prediction)
    rm(actual)
    ok[[i]] <- results



  }
  return(ok)
  stopCluster(cl)
}


mean_and_sd <- function(x) {
  c(round(mean(x, na.rm = TRUE), digits = 2),
    round(sd(x, na.rm = TRUE), digits = 2))
}

RF_train_classification <- function(x) {
  ok <- RF_training_classification(x)
  results <- data.frame(ok)
  rm(ok)
  data <- data.frame(results)
  rm(results)
  m = ncol(data)
  ACC  <- matrix(nrow = m, ncol = 1)
  SENS  <- matrix(nrow = m, ncol = 1)
  SPEC  <-matrix(nrow = m, ncol = 1)
  MCC <- matrix(nrow = m, ncol = 1)

  for(i in 1:m){
    ACC[i,1]  = (data[1,i]+data[4,i])/(data[1,i]+data[2,i]+data[3,i]+data[4,i])*100
    SENS[i,1]  =  (data[4,i])/(data[3,i]+data[4,i])*100
    SPEC[i,1]  = (data[1,i]/(data[1,i]+data[2,i]))*100
    MCC1      = (data[1,i]*data[4,i]) - (data[2,i]*data[3,i])
    MCC2      =  (data[4,i]+data[2,i])*(data[4,i]+data[3,i])
    MCC3      =  (data[1,i]+data[2,i])*(data[1,i]+data[3,i])
    MCC4  =  sqrt(MCC2)*sqrt(MCC3)


    MCC[i,1]  = MCC1/MCC4
  }
  results_ACC <- mean_and_sd(ACC)
  results_SENS <- mean_and_sd(SENS)
  results_SPEC <- mean_and_sd(SPEC)
  results_MCC <- mean_and_sd(MCC)
  rm(ACC)
  rm(SENS)
  rm(SPEC)
  rm(MCC)
  rm(data)
  rm(m)
```

```
  results_all <- (data.frame(c(results_ACC, results_SENS, results_SPEC, results_MCC)))
  rownames(results_all) <- c("ACC_Mean", "ACC_SD", "Sens_Mean", "Sens_SD", "Spec_Mean", "Spec_SD",
                             "MCC_Mean", "MCC_SD")
  rm(results_ACC)
  rm(results_SENS)
  rm(results_SPEC)
  rm(results_MCC)
  names(results_all) <- c("Training Performance")
  return(results_all)
}

RF_testing_classification <- function(x) {
  library(parallel)
  library(doSNOW)
  cl <- makeCluster(8)
  registerDoSNOW(cl)

  output <- list(100)
  output <- foreach(i = 1:100 ) %dopar% {

    data <- x
    active <- subset(data, Activity == "active")
    inactive <- subset(data, Activity == "inactive")
    active <- dplyr::sample_n(active, size =  83, replace = TRUE)
    data <- rbind(active, inactive)
    trainIndex <- sample(nrow(data), size = as.integer(nrow(data) * 0.8),
                         replace = FALSE)
    train <- data[trainIndex, ]
    test <- data[-trainIndex, ]
    model_train <- ranger::ranger(Activity~., data = train, write.forest = TRUE, save.memory = TRUE)
    rm(ctrl)
    rm(rf)
    rm(data)
    rm(trainIndex)
    actual <- test$Activity
    prediction <- predict(model_train, test)
    prediction <- prediction$predictions
    rm(train)
    rm(test)
    rm(model_train)
    results <- caret::confusionMatrix(prediction, actual)
    results <- results$table
    results <- as.numeric(results)
    rm(prediction)
    rm(actual)
    output[[i]] <- results



  }
  return(output)
  stopCluster(cl)
}
```

```r
mean_and_sd <- function(x) {
  c(round(mean(x, na.rm = TRUE), digits = 2),
    round(sd(x, na.rm = TRUE), digits = 2))
}

RF_test_classification <- function(x) {
  ok <- RF_testing_classification(x)
  results <- data.frame(ok)
  rm(ok)
  data <- data.frame(results)
  rm(results)
  m = ncol(data)
  ACC  <- matrix(nrow = m, ncol = 1)
  SENS  <- matrix(nrow = m, ncol = 1)
  SPEC  <-matrix(nrow = m, ncol = 1)
  MCC <- matrix(nrow = m, ncol = 1)

  for(i in 1:m){
    ACC[i,1]  = (data[1,i]+data[4,i])/(data[1,i]+data[2,i]+data[3,i]+data[4,i])*100
    SENS[i,1]  =  (data[4,i])/(data[3,i]+data[4,i])*100
    SPEC[i,1]  = (data[1,i]/(data[1,i]+data[2,i]))*100
    MCC1      = (data[1,i]*data[4,i]) - (data[2,i]*data[3,i])
    MCC2      =  (data[4,i]+data[2,i])*(data[4,i]+data[3,i])
    MCC3      =  (data[1,i]+data[2,i])*(data[1,i]+data[3,i])
    MCC4  =  sqrt(MCC2)*sqrt(MCC3)


    MCC[i,1]  = MCC1/MCC4
  }
  results_ACC <- mean_and_sd(ACC)
  results_SENS <- mean_and_sd(SENS)
  results_SPEC <- mean_and_sd(SPEC)
  results_MCC <- mean_and_sd(MCC)
  rm(ACC)
  rm(SENS)
  rm(SPEC)
  rm(MCC)
  rm(data)
  rm(m)
  results_all <- (data.frame(c(results_ACC, results_SENS, results_SPEC, results_MCC)))
  rownames(results_all) <- c("ACC_Mean", "ACC_SD", "Sens_Mean", "Sens_SD", "Spec_Mean", "Spec_SD",
                             "MCC_Mean", "MCC_SD")
  rm(results_ACC)
  rm(results_SENS)
  rm(results_SPEC)
  rm(results_MCC)
  names(results_all) <- c("Testing Performance")
  return(results_all)
}
```

```
RF_10_CV <- function(x){
  library(parallel)
  library(doSNOW)
  cl <- makeCluster(8)
  registerDoSNOW(cl)

  results <- list(100)
  results <- foreach(i = 1:100 ) %dopar% {
    data <- x
    active <- subset(data, Activity == "active")
    active <- dplyr::sample_n(active, size = 83, replace = TRUE)
    inactive <- subset(data, Activity == "inactive")
    data <- rbind(active, inactive)
    trainIndex <- sample(nrow(data), size = as.integer(nrow(data) * 0.8),
                         replace = FALSE)
    train <- data[trainIndex, ]
    test <- data[-trainIndex, ]
    k = 10
    index <- sample(1:k, nrow(train), replace = TRUE)
    folds <- 1:k
    myRes <- data.frame()
    for (j in 1:k) {
      training <- subset(train, index %in% folds[-j])
      testing <- subset(train, index %in% c(j))
      model_train <- ranger::ranger(Activity~., data = training, write.forest = TRUE, save.memory = TRU
      prediction <- predict(model_train, testing)
      prediction <- prediction$prediction
      actual <- testing$Activity
      ok <- data.frame(prediction = as.character(prediction), actual = as.character(actual))
      myRes <- rbind(myRes, ok)
    }

    prediction <- myRes$prediction
    actual <- myRes$actual
    output <- caret::confusionMatrix(myRes$prediction, myRes$actual)
    rm(myRes)
    output <- output$table
    output <- as.numeric(output)
    results[[i]] <- output
  }
  return(results)
  stopCluster(cl)
}


mean_and_sd <- function(x) {
  c(round(mean(x, na.rm = TRUE), digits = 2),
    round(sd(x, na.rm = TRUE), digits = 2))
}

RF_10_cross_validation <- function(x) {
  ok <- RF_10_CV(x)
  results <- data.frame(ok)
```

```
  rm(ok)
  data <- data.frame(results)
  rm(results)
  m = ncol(data)
  ACC  <- matrix(nrow = m, ncol = 1)
  SENS  <- matrix(nrow = m, ncol = 1)
  SPEC  <-matrix(nrow = m, ncol = 1)
  MCC <- matrix(nrow = m, ncol = 1)

  for(i in 1:m){
    ACC[i,1]  = (data[1,i]+data[4,i])/(data[1,i]+data[2,i]+data[3,i]+data[4,i])*100
    SENS[i,1]  =  (data[4,i])/(data[3,i]+data[4,i])*100
    SPEC[i,1]  = (data[1,i]/(data[1,i]+data[2,i]))*100
    MCC1      = (data[1,i]*data[4,i]) - (data[2,i]*data[3,i])
    MCC2      =  (data[4,i]+data[2,i])*(data[4,i]+data[3,i])
    MCC3      =  (data[1,i]+data[2,i])*(data[1,i]+data[3,i])
    MCC4  =  sqrt(MCC2)*sqrt(MCC3)


    MCC[i,1]  = MCC1/MCC4
  }
  results_ACC <- mean_and_sd(ACC)
  results_SENS <- mean_and_sd(SENS)
  results_SPEC <- mean_and_sd(SPEC)
  results_MCC <- mean_and_sd(MCC)
  rm(ACC)
  rm(SENS)
  rm(SPEC)
  rm(MCC)
  results_all <- (data.frame(c(results_ACC, results_SENS, results_SPEC, results_MCC)))
  rownames(results_all) <- c("ACC_Mean", "ACC_SD", "Sens_Mean", "Sens_SD", "Spec_Mean", "Spec_SD",
                             "MCC_Mean", "MCC_SD")
  names(results_all) <- c("10_Fold_Cross_Validation")
  return(results_all)
}
```

Performance results (Training, Cross Validation, Testing)

```
training <- RF_train_classification(real_data)
training
```

```
##           Training Performance
## ACC_Mean                 96.36
## ACC_SD                    1.06
## Sens_Mean                96.52
## Sens_SD                   1.77
## Spec_Mean                96.14
## Spec_SD                   2.31
## MCC_Mean                  0.93
## MCC_SD                    0.02
```

```
cv <- RF_10_cross_validation(real_data)
cv
```

```
##           10_Fold_Cross_Validation
## ACC_Mean                     91.36
## ACC_SD                        2.40
## Sens_Mean                    90.28
## Sens_SD                       3.21
## Spec_Mean                    92.38
## Spec_SD                       3.03
## MCC_Mean                      0.83
## MCC_SD                        0.05
```

```
testing <- RF_test_classification(real_data)
testing
```

```
##           Testing Performance
## ACC_Mean                90.29
## ACC_SD                   5.13
## Sens_Mean               89.68
## Sens_SD                  8.13
## Spec_Mean               91.48
## Spec_SD                  7.99
## MCC_Mean                 0.81
## MCC_SD                   0.10
```